Using the Domain/Access Application

Order No. 008012
Revision 02

Apollo Computer Inc.
330 Billerica Road
Chelmsford, MA 01824

Contents

Illustrations

Tables

Chapter 1

Overview

With the Domain/Access application, you can transparently access files on Digital Equipment Corporation (DEC*) VAX/VMS computers from your Apollo workstation. You can perform most of the same operations on VAX/VMS files that you can on Domain® files, so the VAX computer appears to be part of the Apollo network. Specifically, you can use the following commands and utilities on VAX/VMS directories and files with the Domain/Access application:

- Most Aegis and UNIX shell commands, such as the Aegis **wd** (working directory) command or the UNIX **cd** (change working directory) command
- Language-level input/output procedures, such as the Pascal **open** and **read** procedures

*DEC is a registered trademark of Digital Equipment Corporation.

- Certain Aegis system calls, such as **ios_$** calls
- Certain Domain/IX system calls, such as **read** and **write**

Because of differences between the Domain and VMS file systems, some commands and procedures have limitations when you use them on VMS files. We describe these differences later in Chapter 3 of this manual.

The Domain/Access application supports operations on all VMS file types. The only exceptions are VMS file types that don't have an equivalent Domain file type. For example, you may copy VMS index files (files with the .idx extension) to your Apollo workstation, but you can't access the index file by a key as you would on the VAX because Aegis doesn't have a similar file type.

Communicating with the VAX

The Domain/Access application relies on several different hardware and software products to communicate with the VAX/VMS computer. On the Apollo side, the Domain/Access application routes data through a gateway node equipped with TCP/IP software and with network controller hardware (refer to *Planning for TCP/IP* for details on appropriate hardware for your network). On the VAX side, data is routed through a VAX/VMS computer equipped with TCP/IP hardware and software. The Apollo gateway node and VAX/VMS computer are connected to an ETHERNET® network. The Domain/Access application makes all of this communications hardware and software transparent to you.

Figure 1–1 shows a typical network configuration that supports the Domain/Access application. For more specific information about the software, hardware, and telecommunications requirements for Domain/Access, refer to *Managing the Domain/Access Application*.

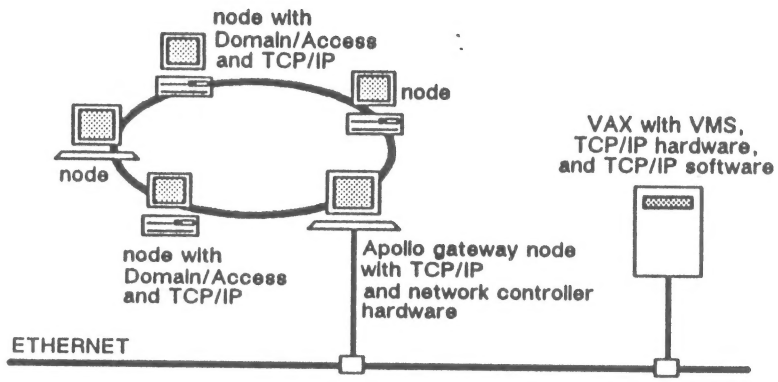

Figure 1–1. A Network Configuration for the Domain/Access Application

Note that the Domain/Access application supports only the VMS or MicroVMS operating systems. It does *not* support access to UNIX operating systems on VAX computers.

Preparing for the Domain/Access Application

Before you can access files transparently with the Domain/Access application, the following steps must be completed by you and your system administrator.

1. Your system administrator must install the TCP/IP software and the appropriate network controller hardware on the gateway node.
2. Your system administrator must install the Domain/Access software on his or her node. Then, he or she must copy parts of the software to the VAX *and* configure the VAX. The system administrator also must create **gateway objects**, which are similar to links, for you to use in pathnames intended for the VAX. These tasks are described in the *Domain/Access Release Document* and *Managing the Domain/Access Application* respectively.

3. You must install on your node the Domain/Access software (described in the *Domain/Access Release Document*) and either the Domain TCP/IP product (described in the *Domain TCP/IP Release Document*) or the Domain/IX BSD4.2 TCP/IP (described in the *Domain/IX BSD4.2 TCP/IP Release Document*).

4. You must provide information that enables access to the VAX/VMS system, as described in Chapter 2 of this manual. Chapter 2 also discusses the standard VAX/VMS file-naming conventions. Appendix A summarizes this information.

Accessing Files on the VAX

After you complete the four steps in the previous section, you can enter commands or execute a program to access the VMS file system, as described in Chapters 3 and 4 of this manual. These chapters list the commands and system calls that the Domain/Access application supports and describe any differences between their operation on Domain files and VMS files.

The Domain/Access software consists of the following parts that work together to access VAX files:

- A **type manager** on each user's Apollo workstation that translates the Domain object types and record formats into the appropriate VAX counterparts and vice versa

- A **gateway object** on an Apollo node (designated by your system administrator) that contains the connection information needed by the type manager to establish a TCP/IP connection with the VAX

- A **server** on the VAX computer that manages the command and system call requests from Domain/Access users

The only change you need to make in accessing a VAX file is to include the name of the gateway object in the pathname of the VAX file. The type manager on your workstation opens the gateway object and uses the information in it to establish a TCP/IP connection with the server on the VAX and logs you into the VAX. Finally, the server accesses the file you requested.

Chapter 2 explains the gateway object in more detail.

Definitions

A glossary of terms and abbreviations is included at the end of this manual, but there are several terms that you should understand before you begin using the Domain/Access product. In this manual we use four similar terms for four similar, but different, concepts. These terms are:

- Filename
- File name
- File specification
- Pathname

A **filename** represents any identifier for a file or directory. A **file name** is a specific component in a file specification. A **file specification** identifies a VAX/VMS file or directory to the VAX/VMS system. See Appendix A for detailed information about VAX/VMS file specifications and file names. A **pathname** identifies a file or directory to the Domain system. The *Domain System User's Guide* describes Domain pathnames.

In addition, we use the term **Domain** to refer to both the Aegis and UNIX operating systems and the term **VAX/VMS** to refer to both VAX/VMS and MicroVAX/MicroVMS operating systems.

Chapter 2

Enabling Access to VAX/VMS Files

Whenever you execute a command or procedure that uses a VAX/VMS file, your workstation must first gain access to the VAX system. For your workstation to do this, you must complete the following steps:

- Ask your system administrator to create a gateway object for you for each VAX system you plan to access.

  For each VAX system you will need to provide your system administrator with your username, password, and (optionally) your account.

- Observe rules when specifying pathnames

  There are overlaps and conflicts between the characters used in VAX/VMS file specifications and the special characters

(or metacharacters) used in various shell commands. Therefore, you must follow a special set of rules to refer to VAX/VMS files from an Apollo workstation.

The rest of this chapter explains these steps in detail.

What is a Gateway Object?

A VMS gateway object is a Domain object that is similar to a link. Thus, the gateway object is known as an Aegis file system object of type *v1_gate* just like an ASCII text file is known as an Aegis file system object of type *uasc*. As mentioned in Chapter 1, a gateway object represents an association, or communications path, between the type manager on your workstation and the VAX you wish to access.

Your system administrator creates one or more gateway objects for you and other users to include in pathnames to VAX files. Ask your system administrator for the names of the gateway objects in your network. (If you want to create your own gateway objects, refer to *Managing the Domain/Access Application* for instructions.)

When you include a gateway object in a pathname, it indicates to the type manager that the pathname refers to a VAX/VMS file. When the type manager opens the gateway object, it finds the following information:

- The Internet name of the VAX node that is the TCP/IP host for the VAX network
- The name of the VAX node that contains the files you wish to access
- The VAX account information necessary to log you in

In some cases, the VAX node that is the TCP/IP host and the VAX node that you want to access are the same node. For example, if you wanted to access vax1 in Figure 2–1, you would be accessing the VAX node that is the TCP/IP host.

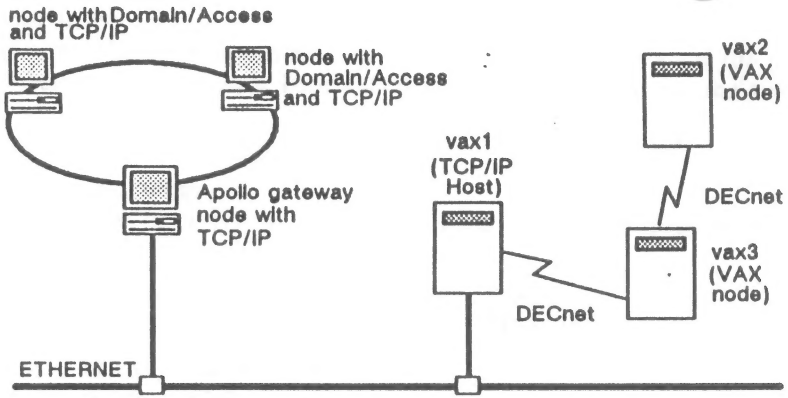

Figure 2–1. Accessing Files on the TCP/IP Host and on Other VAX Nodes

In other cases, you may need to access another VAX node through the DECnet™ network. In Figure 2–1, vax2 and vax3 are examples of VAX nodes on the DECnet* network.

Therefore, if your system administrator is creating gateway objects for the network in Figure 2–1, he or she could create three different gateway objects. Each object would contain the same Internet name (the name of vax1), but different VAX node names (the names of vax1, vax2, and vax3).

In a pathname that refers to a VAX/VMS file, the name of the gateway object separates the Domain names and VMS names. Therefore, you can include only a single VMS gateway in a pathname. For example, the following pathname specifies a VAX/VMS file:

```
//node_a/v_vax2/dr3:/comtests.dir/com2.pas/4
```

where the VMS gateway *v_vax2* indicates the end of the Domain pathname and the beginning of the VMS pathname.

The portion of the pathname to the *left* of the VMS gateway name must meet all Domain file conventions and can use all the features of the Domain file system. For example, you could create a link from your node to the gateway object on another node and then use a relative pathname to refer to the gateway object. For example,

*DECnet is a trademark of Digital Equipment Corporation

in a UNIX shell

```
ln -s //node_a/v_vax2 //my_node/v_vax2
ls /v_vax2/dr3:/comtests.dir
```

or in an Aegis shell:

```
crl //my_node/v_vax2 //node_a/v_vax2
ld /v_vax2/dr3:/comtests.dir
```

The portion of a pathname to the *right* of the gateway object must specify a valid VMS file or directory. However, you don't use standard VMS notation in the VMS filename, but rather a "Domain like" format. The shell would interpret the brackets ([,]) and semicolon (;) used in standard VMS format as special characters, resulting in an error. The "Specifying VAX/VMS Files" section in this chapter completely describes the rules that you must use to refer to VMS filenames in a Domain/Access pathname. In addition, Appendix A briefly describes the standard VAX/VMS file specification conventions.

Figure 2–2 shows the relationships between Domain files, VMS files and gateway objects. The thick lines indicate how the Domain pathname is converted into a VAX/VMS file specification. The thinner lines indicate the communications path used to access the file.

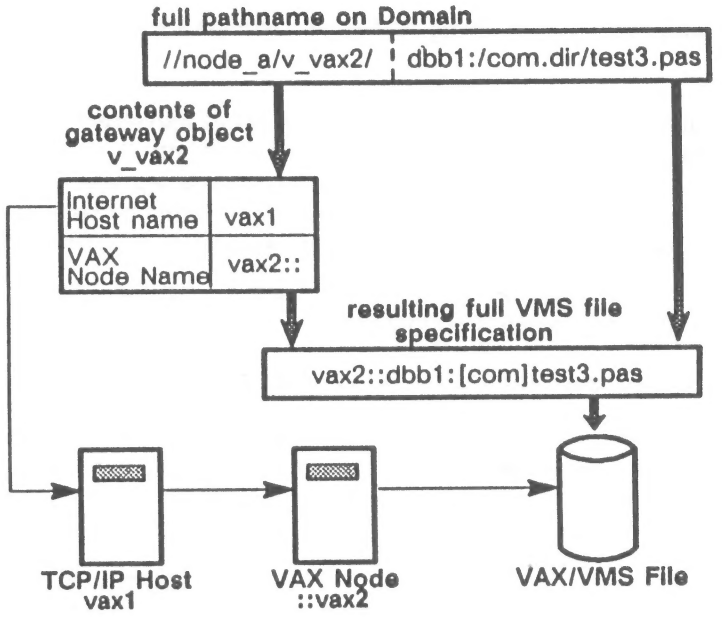

Figure 2–2. The Relationships Between Gateway Objects and VAX/VMS Pathnames

Providing Log–In Information

The Domain/Access application automatically logs in to the VAX/VMS system whenever necessary. However, you must provide it with the information it needs to complete the log–in procedure. That is, you must provide a user ID and, optionally, a password and an account. If you supply an account, you must supply a password also (i.e., you can't use a null password).

The Domain/Access application uses one of two sources for the log–in information, in the following order of priority:

1. The user ID, a password, and an account appended to a pathname
2. The login information associated with the gateway

If you don't know what your log–in information consists of, see your Domain system administrator or your VAX administrator for the information.

Appending Log-In Information to a Pathname

You can include your user ID, password, and account directly in the pathname. To do so, you must follow these guidelines:

- If you're using an Aegis shell, enclose the entire pathname, including the log-in information, in single (') quotation marks.
- If the last item in the pathname is a gateway object, node, device, or directory (i.e., not a file), add a slash after it.
- Put the log-in information immediately after the pathname. A blank is *not* allowed between the pathname and the log-in information.
- Enclose the log-in information in double (") quotation marks.
- Separate the user ID from the password and the password from the account with a blank. You *must* include the password and account if they are required to log in.

Thus, the pathname could look like one of the following:

```
'//jones/v_paris/com2.dir/test.pas"tom tvh mktg"'

'myvax/dm3:/tst.dir/vax.dir/"bob nw72"'

'/vaxgate/dba2:/henri.dir/tewtst.pas/3"leclerc"'
```

In the last case, where you do not provide a password, the Domain/Access software does not send a password.

The guidelines discussed previously assume that you are accessing files on the VAX node that is the TCP/IP host. You only need one user ID, password, and account to access files. However, if you access files on a VAX node that is *not* the TCP/IP host, you may need to append *two* sets of login information.

For example, suppose you need to access the file */dm3:/com2.dir/test.pas* on the VAX node named *vax2*, where the TCP/IP host is called *vax1* and the gateway object is called *v_vax2*. You would append your log-in information for the VAX node (vax2) *and* for the TCP/IP host (vax1) respectively, *without* a space separating the two sets of log-in information. Therefore, the pathname would look similar to the following:

```
'/v_vax2/dm3:/com2.dir/test.pas"tom tvh
mktg""tomh tvh sales"'
```

The only exception to this requirement for two sets of log-in information would be the case where your log-in information for both vax1 and vax2 is *identical*. In that case, you append one set of log-in information to the pathname.

Up to this point, our examples have assumed that you access files from the VAX node that is named in the gateway object. If your VAX network uses DECnet, you can access other VAX nodes without a separate gateway object that points to each VAX node. For example, suppose you want to access a file with the same name as the one in the previous example, but in this case, the file resides on *vax3*, which is connected to vax1 and vax2 via DECnet. You could access *vax3* with the same gateway object, *v_vax2*, as follows:

```
'/v_vax2/vax3::/dm3:/com2.dir/test.pas"tom tvh
mktg""tomh tvh sales"'
```

Specifying VAX/VMS Files

Before you read this section you should be familiar with both Domain pathnames and VAX/VMS file specifications. See the *Domain System User's Guide* for information on Domain pathnames. See Appendix A for an introduction to VMS file specifications.

You name VAX/VMS files by appending a VMS filename to the name of a VMS gateway object. For example, assume your system administrator created *//node_d/v_paris* as a VMS gateway object for the VAX/VMS system with the node name *paris::*. You can now enter the following command to list the files contained in the *[teletest]* directory on the dm2: device on the node *paris::*

```
in a UNIX shell
ls //node_d/v_paris/dm2:/teletest.dir

in an Aegis shell
ld  //node_d/v_paris/dm2:/teletest.dir
```

The left part of the pathname and the gateway object name must meet standard Domain naming conventions. For example, you can use the naming directory (~) or the parent directory (\) symbols in the Domain part of the pathname.

The part of the pathname to the right of the VMS gateway name corresponds to the VAX/VMS device name and directory name; however, we use a Domain format syntax. We use this syntax because the DM and the shells often apply special interpretations to the brackets and semicolons that are also used in the VMS file-naming syntax. The following sections discuss how to avoid such conflicts in VMS filenames.

Character Conflicts

Most software that interprets commands, such as the DM and the Aegis, Bourne, and C shells include command line parsers and other facilities that interpret certain characters as special, or metacharacters. Many of the nonalphanumeric characters that you use to construct VAX/VMS file specifications have such special meanings when used in Domain commands.

Table 2-1 lists the nonalphanumeric characters that are valid in VMS file specifications and indicates whether each is a special character for the DM, Aegis shell, C shell, and Bourne shell. (Check the appropriate documentation for any other cases.)

You must use one (or both) of the following techniques to prevent VMS file specification characters from being interpreted as command special characters:

- Use Domain format in the VMS file specification.
- Escape the conflicting VMS file specification characters so that they are not interpreted by the node software.

Table 2-1. Special Character Use

Character	Special Character In			
	DM	Aegis	C	Bourne
:	N	N	N	N
.	N	N	N	N
[	Y	Y	Y	Y
]	Y	Y	Y	Y
;	Y	Y	Y	Y
$	N	N	Y	Y

Domain Pathname Format for VAX/VMS Files

You can avoid conflicts between VAX/VMS file specification characters and special characters by specifying the VMS file in a Domain pathname format. In this format, you replace certain VMS file specification characters with their Domain equivalents.

To convert a VAX/VMS file specification to Domain format, complete the following steps:

1. Replace *each* of the following characters in the VAX/VMS file specification with a slash (/), the standard Domain pathname component separator:

   [At the start of the directory specification

   ] At the end of the directory specification

   ; Before the file version number

   . Between directory components *only*

2. Append *.dir* to each directory component.

The following rules also apply to the Domain pathname format:

- You *cannot* use Domain pathname format to replace the $ and : characters. When you use the C shell or Bourne shell, you must escape (or enclose in quotation marks) the $ character independent of the pathname format you use.
- You *must not* replace the period that separates the file name from the extension.

For example, to print the VMS file

```
paris::dma3:[com.tests]echo.pas;3
```

using the VMS gateway */node_b/v_paris* and Domain pathname format, enter the command:

for a UNIX shell
```
lpr //node_b/v_paris/dma3:/com.dir/tests.dir/echo.pas/3
```

for an Aegis shell
```
prf //node_b/v_paris/dma3:/com.dir/tests.dir/echo.pas/3
```

Finally, if you include the name of a DECnet node in a pathname, use the name plus ::. For example, if you wanted to access a file on *london::*, which is connected to *paris::* from the previous example, you'd use a pathname similar to the following:

```
//node_b/v_paris/london::/dbb1:/tests.dir/
test3.pas.3
```

Escaping Special Characters

You can prevent a shell from interpreting special characters by escaping them. The shell then accepts the escaped characters as literal parts of the pathname. Each shell or command interpreter has its own rules for escaping characters.

Aegis shell and DM

To escape special characters when you use the Display Manager or the Aegis shell, precede each special character with the at sign (@) escape character. For example:

```
//node_b/v_paris/dma3:@[com.tests@]echo.pas@;3
```

You can also use single quotation marks in DM commands to escape a complete pathname. However, the single quotation marks *do not* escape VMS pathname special characters in shell commands. In addition, the double quotation marks *do not* escape VMS pathname special characters in either DM or shell commands.

Bourne and C Shells

To escape special characters when you use the Bourne shell or C shell, precede each special character with a backslash (\) or enclose the pathname in single quotation marks. For example:

```
//node_b/v_paris/dma3:\[com.tests\]echo.pas\;3
```

or

```
'//node_b/v_paris/dma3:[com.tests]echo.pas;3'
```

To include log-in information in a quoted Bourne or C shell pathname, nest the quotation marks. You can use either single or double quotation marks around the log-in information. For example, use the following pathname to access the file *paris::dmc1:[sean]tax.pas;5* using the log-in ID *sean* and the password *secret*:

```
'//node_b/v_paris/dmc1:[sean]tax.pas;5"sean
secret"'
```


Wildcards in VAX/VMS Pathnames

You can use wildcards in VAX/VMS pathnames. Follow the standard wildcarding procedures that apply to the shell that you are using. If you use wildcards, you *must* use the Domain pathname format for specifying VAX/VMS files instead of escaping special characters.

NOTE: Do *not* use VAX/VMS wildcards. Use only Domain wildcards.

For example, the following command lists all versions of all files in *paris::dba3:[teletest]* that start with the letters "com." Note that the command uses the Domain pathname format as described above.

```
in a UNIX shell
ls //smith/v_paris/dba3:/teletest.dir/com?*
in an Aegis shell
ld //smith/v_paris/dba3:/teletest.dir/com?*
```

Using Relative Pathnames

The Domain/Access application lets you use relative pathnames to refer to VMS files as you would use them for Domain files. However, when you log directly onto a VAX/VMS system, the initial pathname defaults for VMS files are different than the defaults you're used to with the Domain system. If you list the contents of the gateway object directory, the VAX defaults to your home directory on the VAX. For example,

```
in a UNIX shell
ls  /v_vax2
[my_home_dir]

memo.txt        ltr.txt
proj.dir

in an Aegis shell
ld  /v_vax2
[my_home_dir]

memo.txt        ltr.txt
proj.dir
```

Suppose you then decide to list the contents of the *proj.dir* directory by typing the following relative pathname:

in a UNIX shell
```
ls /v_vax2/proj.dir
```

in an Aegis shell
```
ld /v_vax2/proj.dir
```

In this case, the VAX defaults to the default device, the directory that contains your home directory (one directory above your home directory). Since *proj.dir* is in your home directory, not in the default device directory, you will receive an error (proj.dir not found). Therefore, using relative pathnames without specifying your default device and home directory first can possibly cause confusion.

We suggest that you initially specify the device and directory name in the VMS portion of a pathname before you begin using relative pathnames. Because the Domain/Access application supports working directory commands (wd or cd) on VAX/VMS files, you can set your working directory to any VAX/VMS directory and then access files within that directory with relative pathnames.

For example, assume your working directory is *//node_a/reni/doc*, and you want to work on files in the *dbb3:[karla.comtest]* directory on the VMS node *paris::*. If *//jones/v_paris* is the VMS gateway object for *paris::*, you can set your working directory with the command:

in a UNIX shell
```
cd //jones/v_paris/dbb3:/karla.dir/comtest.dir
```

in an Aegis shell
```
wd //jones/v_paris/dbb3:/karla.dir/comtest.dir
```

You can now print the file *com1.pas* in this directory by entering:

in a Unix shell
```
lpr com1.pas
```

in an Aegis shell
```
prf com1.pas
```

Note that the prf command prints the file at your local printer (the default /sys/print printer).

Chapter 3

Accessing Files with Commands

The Domain/Access application enables you to access VAX/VMS files through Display Manager (DM) commands and through Aegis and UNIX shell commands.

No matter what commands you use, you can only access VAX/VMS files with up to 2,147,483,647 (or 2^{31}) records. Each record can be up to 4094 bytes long.

Using the Display Manager

You can use the Display Manager (DM) on VAX/VMS files the same way you use it for files that are located on Apollo workstations. Once you have enabled access to the VAX system as described in Chapter 2, DM commands have the same effect on VMS and Domain files. You *must* follow the file-naming rules described in Chapter 2 when you refer to VAX/VMS files.

NOTE: You *cannot* use DM commands to create or manage processes on the VAX system. (These commands include **cp** (create process, pads, and windows), **cpo** (create process without pads or windows), **cps** (create process independent of login), **dc** (continue a suspended process), **dq** (generate a quit fault in a process), **ds** (suspend a process), and **shut** (shut down system).)

Using Aegis Shell Commands

The Domain/Access application allows you to use most of the standard Aegis shell commands on VAX/VMS files. However, because of differences between the Aegis file system and the VMS file system, a few shell commands produce slightly different results when used on VMS files.

Command Line Special Characters

You can use all Aegis shell command line special characters in commands that refer to VAX/VMS files. These include:

- Aegis pathname wildcards
- Input/output control characters
- Parsing operators

For example, the following command:

```
$ catf  /vx1/dm0:/jo.dir/tst?.txt  >  /vx1/dm0:/jo.dir/tsts.txt
```

combines several files and writes the result to the VAX. If there are files in the jo.dir directory file on the VAX named *tst1.txt*, *tst2.txt*, and *tst3.txt*, the "?" wildcard causes all three files to be catenated (that is, read and written). Because the command also uses the ">" redirection character, the results are written to the single VMS file *dm0:[jo]tsts.txt*.

Note that in this example the pathname uses Domain format for the VMS file specifications. Chapter 2 describes these two techniques in detail. See the *Domain System Command Reference* for detailed information on Aegis shell special characters.

Aegis Shell Command Support

Table 3-1 lists the Aegis shell commands, and indicates whether they are fully supported, unsupported, or supported with restrictions on VAX/VMS files. The sections that follow this table describe the restrictions on each call that is marked in this table with an R. See the *Domain System Command Reference* for details on how to use the commands.

Note that this list does not include commands that do not access user files, such as process management commands, or commands that manipulate files or objects that permanently reside on Apollo systems, such as Aegis system files or objects. The Domain/Access product doesn't support these commands

Table 3-1. Aegis Shell Command Support

Command	Support Y/N/R	Description
arcf	N	Maintain an archive file
args	Y	Echo command line arguments
acl	N	List or copy an access control list
bind	N	Combine object modules
catf	Y	Catenate (copy) files to standard output
cc	R	Compile a C program
chhdir	N	Change the login home directory
chn	Y	Change an object's name
chpat	Y	Replace a pattern in a text file
chuvol	N	Change a volume's UID
cmf	Y	Compare ASCII files; identify differences

Table 3-1. Aegis Shell Command Support (Continued)

Command	Support Y/N/R	Description
cmsrf	Y	Compare sorted files for common lines
cmt	N	Compare source tree to target tree
cpboot	N	Copy an Aegis Bootstrap file
cpf	R	Copy a file or files
cpl	R	Copy a link
cpscr	N	Copy the screen display to a file
cpt	Y	Copy a directory tree
crd	Y	Create a directory
crefs	Y	Cross reference symbols in a file
crf	N	Create a file
crl	R	Create a link
crpad	N	Create a transcript pad and window
crsubs	N	Create a protected subsystem
ctob	N	Catalog an object
cvt_rec_usac	N	Convert file types
db	N	Low level debugger
dcalc	Y	Desk calculator emulator
dbug	N	Language level debugger
dldupl	Y	Delete duplicated lines from a file
dlf	R	Delete one or more files
dll	R	Delete a link
dlt	R	Delete a tree
dmtvol	N	Dismount a logical volume
ed	Y	Line editor

Table 3-1. Aegis Shell Command Support (Continued)

Command	Support Y/N/R	Description
edacl	N	Edit or list an access control list
edfont	N	Edit a character font
edmtdesc	N	Edit a magnetic tape descriptor file
edstr	Y	Edit a stream
emt	Y	Emulate a dumb terminal
ensubs	N	Enter a protected subsystem
exfld	Y	Manipulate data in formatted fields
find_orphans	N	Locate and catalog uncataloged objects
flen	Y	Count lines, words, and chars in a file
fmc	Y	Format text into multiple columns
fmt	Y	Format a text file
fpat	Y	Find a text pattern in an ASCII file
fpatb	Y	Find blocks of text containing patterns
fserr	Y	Find spelling errors
ftn	R	Compile a FORTRAN program
hpc	N	Program counter histogram
lamf	Y	Laminate files to standard output
lbr	N	Compile object modules into a library
ld	R	List contents of a directory
lkob	N	Lock an object
llkob	N	List locked objects
lvolfs	N	List free space on a logical volume
macro	Y	Expand macro definitions

Table 3-1. Aegis Shell Command Support (Continued)

Command	Support Y/N/R	Description
mtvol	N	Mount a logical volume
mvf	Y	Move a file
nd	Y	Set or display the naming (-) directory
obty	R	Set or display the type of an object
old_ edfont	N	Edit a character font
os	Y	Convert ASCII to FORTRAN carriage control
pagf	Y	Paginate a file
pas	R	Compile a Pascal program
prf	Y	Queue a file for printing
prfd	Y	Invoke menu-based print file utility
rbak	N	Restore or index a backup file
revl	Y	Reverse each line in a file
rwmt	N	Read/write nonDomain magnetic tapes
salacl	N	Salvage an access control list
sald	N	Salvage a directory
sh	N	Invoke a shell (cmd. line interpreter)
siorf	N	Receive a file from a serial I/O line
siotf	N	Transmit a file over a serial I/O line
srf	Y	Sort and/or merge text files
subs	N	Set or display subsystem attributes
tee	Y	Copy input to std. output and to files
tlc	Y	Replace (transliterate) characters

Table 3-1. Aegis Shell Command Support (Concluded)

Command	Support Y/N/R	Description
uctob	N	Uncatalog pathname, don't delete object
ulkob	N	Unlock an object
vt100	N	Emulate a VT100 terminal
wbak	N	Create a backup file
wd	Y	Set or display the working directory
xdmc	N	Execute a DM command from the shell
xsubs	N	Run shell script subsystem manager

NOTES: Several of the unsupported commands manipulate object modules. You can execute or modify Domain object code only if it resides on a Apollo node. For example, you cannot copy an Aegis command from the /com directory onto a VAX/VMS system and then execute it from your node.

Support of **dcalc** and **emt** indicates you may execute command files for these facilities that reside on the VAX/VMS system.

Aegis Commands with Restrictions

The following sections describe limitations that affect the commands that you can execute on VAX/VMS files.

Copying and Deleting Files: cpf and dlf

If you do not specify a file version number, the Domain/Access application always operates upon the highest numbered version of a VAX/VMS file. This default sometimes has an unexpected side effect; for example, a command deletes only a single file, not all versions of the file.

For example, suppose you do the following:

1. You use the **cpf** command with the [*-r*] option to copy a file to the VAX.

2. You do not specify a version number.

3. You don't check to see that two or more versions of the file exist.

You receive a "file already exists" error message, even though you specified *-r*. The following paragraphs explain why this happens.

Next, suppose you have two versions of the file *doe/dba0:/test.dir/comm.pas*. These versions are *comm.pas;4* and comm.pas;3. Suppose you use the following command:

```
$ cpf //my/test/comm.pas /doe/dba0:/test.dir/comm.pas -r
```

The Domain/Access software deletes *comm.pas;4* because of the [*-r*] option, but *comm.pas;3* still exists, and therefore *comm.pas* still exists. **cpf** will not copy a file to an existing file (that is, *comm.pas*), so the copy command returns the error message.

To prevent this problem, either specify the latest version number in your command, or first delete all old versions of a file.

Finally, when you copy a Domain *uasc* file to the VAX, the file becomes a *v1_varrec* file. However, if you copy a *v1_varrec* file back to the Domain system, the file becomes a *records* file. Therefore, you should change the object type of the file back to *uasc* (with the obty shell command) or adjust your operations on the file to reflect the new object type.

Deleting Directories: dlt

The Domain/Access application does not impose restrictions on the deletion of directories. However, most VAX/VMS systems by default do not give you deletion rights on directories. You must use the VAX/VMS DCL **set protection** command to change the directory protection before you delete it. To run the **set protection** command from your Apollo workstation, log in to the VAX system using the telnet program. (Refer to *Using telnet and ftp* for more information about telnet.)

Compiling Programs: cc, ftn, and pas

The source input to the Apollo language compilers, **cc**, **ftn**, and **pas** can be on a VMS system. However, the binary output file must be written to an Apollo node. Therefore, you *must* specify a binary output file in the compiler command if the source file is on a VMS system.

For example, the following command compiles a Pascal source file that is located on a VMS system and writes the binary file to an Apollo node:

```
$ pas /v_boston/db0:/egil.dir/test -b //node_b/vms/test
```

Linking Objects: cpl, crl, and dll

Because the VMS system does not support links, you cannot create links from VMS directories or files to Domain directories and files. However, you can create, copy or delete links that point from Domain directories to VMS directories or files.

In the following example, a valid command creates a link from the Domain directory *//node_a/vax/com/tests* to the VMS directory *paris::db0:[com.tests]*. In this case, the working directory is *//node_a/vax* and the Domain VMS gateway */v_paris* corresponds to the VAX node *paris::*.

```
$ crl  com/tests  /v_paris/db0:/com.dir/tests.dir
```

The inverse of this command, attempting to create a link from the VAX to a Domain directory is *not valid*. For example:

```
$ crl  /v_paris/db0:/com.dir/tests.dir  com/tests
error creating link "v_paris/db0:/com.dir/tests.dir" -
branch is not a directory (os/naming server)
```

Listing Directories: ld

Because of the differences between the VMS file-management system and the Aegis object-management system, the following arguments to an **ld** command will not display the requested information:

-attr	Display permanent/immutable/trouble flags
-ll	List link names
-lt	Link trace; display link name resolution
-r	Display your access rights

If you use the [*-a*] option of the **ld** command, the timestamp of each object will be listed (along with other attributes of each object). Because of differences in the way the Domain and VAX computers interpret time zones, the timestamps will be incorrect. If the Domain and VAX computers are set to the same time zone, the timestamps

will be off by the time zone offset. The time zone offset is the difference in time between Greenwich mean time (GMT) and the current time zone (for example, eastern standard time (EST)).

Setting Object Types: obty

The **obty** command can show only the object type of VAX/VMS files and directories; it cannot set the object type. Therefore, you cannot use the object_type argument if you specify a VAX/VMS file.

Using UNIX Commands

The Domain/Access application allows you to use almost all of the standard UNIX file access and manipulation commands on VAX/VMS files. However, the following limitations do apply:

- UNIX commands cannot create, execute, or access the contents of Domain object modules on the VAX/VMS system. Therefore, you cannot execute the following commands on VAX/VMS files:

  ld (link editor)
  nm (print name list)
  size (size of an object file)

- The output of the **cc** compiler must be sent to an Apollo node, and not to a VAX/VMS file (although the source files can be on the VAX/VMS system).

- The **chmod** (change mode) and **df** (disk free) commands are not supported.

- You need system privileges to delete a directory on the VAX. Therefore, you can't use the **rmdir** (remove (unlink) directories) command without these privileges.

- You can't create a hard link using the **ln** command. (However, you may create a soft link from the Domain to the VAX with the [*-s*] option of the ln command.)

You can use each UNIX shell's standard wildcards in VMS filenames. However, you must then separate components of the VAX/VMS filename using slashes (/). If you use shell metacharacters as part of the VMS filename (for example brackets to indicate a VAX/VMS directory), you must enclose the metacharacters in quotation marks. Chapter 2 discusses file naming in detail.

See the *Domain/IX Command Reference for BSD4.2* for details on how to use BSD4.2 commands. See the *Domain/IX Command Reference for System V* for details on how to use System V commands.

Chapter 4

Accessing Files from Your Program

The Domain/Access application enables you to access VAX/VMS files through the normal programmatic file-access techniques. You can use:

- The standard file access and I/O procedures that are part of your programming language; for example, the Pascal **readln** or **put** procedures. The Domain/Access software supports the standard I/O procedures of Domain C, FORTRAN, and Pascal.
- The Aegis stream and name management system calls. These include **ios_$**, **stream_$**, and certain **name_$** calls.
- UNIX system calls. (These are identical to the calls supplied by the Domain C libraries.)

NOTE: This chapter lists calls and procedures that access and manipulate files and directories. It includes calls that take file or directory names as input or output arguments. It does not include calls that can only manipulate Domain system files or objects.

Standard I/O Procedures

This section describes the Pascal I/O procedures, FORTRAN I/O statements, C library functions, and UNIX system calls that the Domain/Access software supports.

Domain Pascal I/O Procedures

The Domain/Access application fully supports the standard predeclared Domain Pascal input/output procedures, which are listed below. See the *Domain Pascal Language Reference* for details about these procedures and their use.

- **close**
- **find**
- **get**
- **open**
- **page**
- **put**
- **read**
- **readln**
- **replace**
- **reset**
- **rewrite**
- **write**
- **writeln**

Domain FORTRAN I/O Statements

In addition, the Domain/Access application fully supports the standard Domain FORTRAN input/output statements, which are listed below. See the *Domain FORTRAN Language Reference* for details about these procedures and their use.

- **backspace**
- **close**
- **endfile**
- **inquire**
- **open**
- **print**
- **read**
- **rewind**
- **write**

Domain C Library Functions and UNIX System Calls

The Domain/Access application supports both the BSD4.2 and System V Domain C libraries, including the standard library I/O functions and system-level I/O functions.

The Domain/Access software also supports the UNIX system I/O calls. These calls are identical to the functions that are included in the Domain C library. Therefore this section covers both the C library and the UNIX Calls.

Note that the Domain/Access application translates all of the C calls to **ios_$** calls and that the C calls default to unregulated access, which the Domain/Access application ignores. (The Domain/Access software will open the file with regulated access instead; thus, you are limited to one open stream per file.)

Table 4-1 lists standard library I/O (and I/O-related) functions and calls and the system-level library functions and calls. This table indicates whether each call is included in BSD4.2, System V, or both. It also indicates whether each call is fully supported (Y), unsupported (N), or supported with restrictions (R). See the *Domain C (CLIB) Reference* for details on these functions and their use.

Table 4-1. C Function and UNIX Call Support

Function	System		Support Y/N
	BSD4.2	System V	
Standard Library Functions and Calls			
clearerr	✓	✓	Y
ctermid		✓	Y
fclose	✓	✓	Y
fdopen	✓	✓	Y
feof	✓	✓	Y
ferror	✓	✓	Y
fflush	✓	✓	Y
fgetc	✓	✓	Y
fgets	✓	✓	Y
fileno	✓	✓	Y
fopen	✓	✓	Y
fprintf	✓	✓	Y
fputc	✓	✓	Y
fputs	✓	✓	Y
fread	✓	✓	Y
freopen	✓	✓	Y
fscanf	✓	✓	Y
fseek	✓	✓	Y

Table 4-1. C Function and UNIX Call Support (Continued)

Function	System		Support Y/N
	BSD4.2	System V	
Standard Library Functions and Calls (Continued)			
ftell	✓	✓	Y
fwrite	✓	✓	Y
getc	✓	✓	Y
getchar	✓	✓	Y
gets	✓	✓	Y
getw	✓	✓	Y
perror	✓	✓	Y
printf	✓	✓	Y
putc	✓	✓	Y
putchar	✓	✓	Y
puts	✓	✓	Y
putw	✓		Y
rewind	✓	✓	Y
scanf	✓	✓	Y
setbuf	✓	✓	Y
setbuffer	✓		Y
setlinebuf	✓		Y
setvbuf		✓	Y
sprintf	✓	✓	Y
sscanf	✓	✓	Y

Table 4-1. C Function and UNIX Call Support (Continued)

Function	System		Support
	BSD4.2	System V	Y/N
Standard Library Functions and Calls (Continued)			
vprintf	✓	✓	Y
vsprintf	✓	✓	Y
tempnam		✓	Y
tmpfile		✓	Y
tmpnam		✓	Y
ungetc	✓	✓	Y
vfprintf	✓	✓	Y
System Functions and Calls			
access	✓	✓	Y
chdir	✓	✓	Y
chmod	✓	✓	N
chown	✓	✓	N
close	✓	✓	Y
closedir	✓		Y
creat	✓	✓	Y
default_acl	✓	✓	N
dup	✓	✓	Y
dup2	✓		Y
execl	✓	✓	N
execle	✓	✓	N
execlp	✓	✓	N
execv	✓	✓	N
execve		✓	N

Table 4-1. C Function and UNIX Call Support (Continued)

Function	System		Support
	BSD4.2	System V	Y/N
System Functions and Calls (Continued)			
execvp	✓	✓	N
fcntl	✓	✓	Y
flock	✓		N
fstat	✓	✓	Y
fsync	✓		Y
getcwd		✓	Y
ioctl	✓		Y
link	✓	✓	N
lseek	✓	✓	Y
lstat	✓		R
mkdir	✓		Y
mknod	✓	✓	Y
mktemp	✓	✓	Y
open	✓	✓	Y
opendir	✓		Y
read	✓	✓	Y
readdir	✓		Y
readlink	✓		R
readv	✓		Y
rename	✓		Y
rewinddir	✓		Y
rmdir	✓		Y

Table 4-1. C Function and UNIX Call Support (Concluded)

Function	System		Support
	BSD4.2	System V	Y/N
System Functions and Calls (Continued)			
seekdir	✓		Y
soft_link	✓	✓	R
soft_unlink	✓	✓	R
stat	✓	✓	Y
symlink	✓		R
telldir	✓		Y
truncate	✓		Y
umask	✓	✓	Y
unlink	✓	✓	Y
utime	✓	✓	N
utimes	✓		N
write	✓	✓	Y
writev	✓		Y

Domain C and UNIX Call Restrictions

All Domain C functions and UNIX calls that manage symbolic links have limitations on how they are used because the VAX/VMS system does not support symbolic links. As a result, you *cannot* establish a symbolic link from one VMS file or directory to another. However, you can establish a symbolic link from a Domain object to a VAX/VMS file or directory.

For example, the following C function is valid because it links a Domain filename to a particular VAX/VMS file. (In this example */v_paris* is the VMS gateway object for the required VAX node. We show the function on two lines because of this book's margins.)

```
status = symlink(/v_paris/db0:/tsts.dir/tst1.pas,
                 //node_a/vaxtsts/tst1.pas)
```

However, the following C function returns an error status because it attempts to make a link from a VAX/VMS directory. (We show the function on two lines because of this book's margins.)

```
status = symlink(/v_paris/db0:/tsts.dir/tst1.pas,
                 /v_paris/db2:/myststs.dir/tst1.pas)
```

This restriction on links applies to the following functions and calls:

- **lstat**
- **readlink**
- **soft_link**
- **soft_unlink**
- **symlink**

Aegis System Calls

The Domain/Access software supports Aegis **name_$**, **ios_$**, and **stream_$** system calls for accessing and modifying VAX/VMS files and directories. (We suggest that you use **ios_$** calls in new programs because these calls supersede the **stream_$** calls.) However, several functions cannot be fully supported due to the differences in the VMS system. Table 4-2 lists these calls and indicates whether each call is fully supported (Y), unsupported (N), or supported with restrictions (R).

See the *Domain System Call Reference (Volume II)* for details on each call. See *Programming with General System Calls* for information on how to perform stream I/O with **ios_$** calls.

Table 4-2. Aegis System Call Support

Call	Support Y/N/R
ios_$ Calls	
ios_$change_path_name	R
ios_$close	Y
ios_$create	R
ios_$delete	Y
ios_$dup	Y
ios_$equal	Y
ios_$force_write_file	Y
ios_$get	Y
ios_$get_dir	Y
ios_$get_handle	Y
ios_$inq_byte_pos	Y
ios_$inq_conn_flags	Y
ios_$cur_rec_len	Y
ios_$inq_file_attr	Y
ios_$inq_full_key	Y
ios_$inq_mgr_flags	Y
ios_$inq_path_name	Y
ios_$inq_obj_flags	Y
ios_$inq_rec_pos	Y
ios_$inq_rec_remainder	Y
ios_$inq_rec_type	Y
ios_$inq_short_key	Y

Table 4-2. Aegis System Call Support (Continued)

Call	Support Y/N/R
ios_$ Calls (Continued)	
ios_$inq_type_uid	Y
ios_$locate	R
ios_$open	R
ios_$put	Y
ios_$replicate	Y
ios_$seek	Y
ios_$seek_full_key	Y
ios_$seek_short_key	Y
ios_$seek_to_bof	Y
ios_$seek_to_eof	Y
ios_$set_conn_flag	R
ios_$set_dir	Y
ios_$set_locate_buffer_size	Y
ios_$set_obj_flag	R
ios_$set_rec_type	R
ios_$switch	Y
ios_$truncate	Y
name_$ Calls	
name_$add_link	R
name_$cname	N
name_$create_directory	N
name_$create_file	N

Table 4-2. Aegis System Call Support (Continued)

Call	Support Y/N/R
name_$ Calls (Continued)	
name_$delete_directory	N
name_$delete_file	N
name_$drop_link	R
name_$extract_data	N
name_$get_ndir	Y
name_$get_path	Y
name_$get_wdir	Y
name_$read_dir	N
name_$read_link	N
name_$set_ndir	Y
name_$set_wdir	Y
stream_$ Calls	
stream_$close	Y
stream_$create	R
stream_$create_bin	R
stream_$create_here	R
stream_$delete	Y
stream_$force_write_file	Y
stream_$get_buf	Y
stream_$get_conditional	Y
stream_$get_ec	Y
stream_$get_prior_rec	Y
stream_$get_rec	Y

Table 4-2. Aegis System Call Support (Concluded)

Call	Support Y/N/R
stream_$ Calls (continued)	
stream_$inquire	R
stream_$open	Y
stream_$put_chr	Y
stream_$put_rec	Y
stream_$redefine	R
stream_$replace	Y
stream_$seek	Y
stream_$truncate	Y

Aegis Call Restrictions

The following sections describe the restrictions on each Aegis call that has limitations when applied to VAX/VMS files.

ios_$change_path_name Restrictions

The new pathname must conform either to standard VAX/VMS file specification syntax or to the syntax described in the "Domain Pathname Format for VAX/VMS Files" section of Chapter 2. You *cannot* use standard Domain file syntax, and the name must conform to all VMS-imposed restrictions when converted to VAX/VMS syntax. For example, the following pathnames are valid:

```
dma3:[com.tests]echo.pas;3
dma3:/com.dir/tests.dir/echo.pas/3
```

However, the following pathnames are *not* valid:

```
dma3:/com.dir/echo.new.pas/3
dma3:/com.tests/echo.pas/3
```

ios_$create Restrictions

You may specify any valid object type, but the Domain/Access software will create the type according to Table 4-3.

Table 4-3. Domain/Access Object Types and ios_$create

Type Requested	Type Created	Description
directory_$uid	v1_dir_$uid	A VAX/VMS directory
uasc_$uid	v1_varrec_$uid	A VAX/VMS variable length record file
undef_$uid	v1_varrec_$uid	"
records_$uid	v1_varrec_$uid	"
uid_$nil	v1_varrec_$uid	"
Any other type	None. Returns the ios_$ill_obj_type error.	Error

If you create an empty *v1_varrec* file and set the record type to either *ios_$explicit_f2* or *ios_$f1*, the object type will change to *v1_fixrec* (a VAX/VMS fixed length record file.) Further, note that the *mbx_$uid* object type is not supported because **mbx_$** calls are not supported.

An attempt to set an open option of *ios_$unregulated_opt* is ignored.

ios_$open Restrictions

An attempt to set an open option of *ios_$unregulated_opt* is ignored.

ios_$set_conn_flag Restrictions

Only file access is allowed. Therefore, the following connection flags cannot be set to on; any attempt to set them will result in an *ios_$illegal_operation* error:

tty
ipc
vt

ios_$set_obj_flag Restrictions

An attempt to set an object flag of *ios_$of_sparse_ok* is ignored.

ios_$set_rec_type Restrictions

You can use Domain/Access to access any VAX/VMS file with up to 2,147,483,647 (or 2^{31}) records. Each record may be up to 4094 bytes long.

We support the operations on files that are listed in Table 4-4, where Y = supported, N = not supported, and FE = supported on empty files only.

Table 4-4. Domain/Access Record Types

FROM Record Type	TO Record Type			
	ios_$v1	ios_$f2	ios_$explicit_f2	ios_$f1
ios_$v1	Y	N	N	N
ios_$f2	N	N	N	N
ios_$explicit_f2	FE	N	Y	Y
ios_$f1	FE	N	Y	Y

From the table, you can see that we don't support the *ios_$f2* record type. Use the *ios_$v1* record type instead. Also, notice that when you specify the *ios_$explicit_f2* record type, you'll receive the *ios_$f1* record type. (The VMS operating system doesn't have a record type that has fixed records with counts.)

name_$add_link and drop_link Restrictions

You cannot create links on the VAX/VMS system. However, a VMS directory or file can be the target of a link on a Domain workstation.

stream_$create, create_bin, create_here, open Restrictions

You can specify a concurrency of *stream_$unregulated*, but the Domain/Access software will ignore it.

stream_$redefine Restrictions

The following restrictions apply to the attributes that you can set by using the **stream_$redefine** call for a VAX/VMS directory or file.

Flag1

You can reset the record type (Bits 10,11), as long as the type stays within the limitations of the **ios_$set_rec_type** call.

You can specify *stream_$unregulated* concurrency at open (Bits 18, 19), but the Domain/Access software ignores it.

Object Types

You cannot redefine the object type.

Sparse Flag

You can set the Sparse Flag true, but the Domain/Access software ignores it. (No error is reported.)

Programmer's Notes

Because of the way VAX/VMS systems manipulate files, and because of the way the Domain/Access application is implemented, there are several considerations that affect program performance. The following notes list some information that you should keep in mind when writing programs that access VAX/VMS files.

- The Domain/Access type manager uses the following types:

v1_dir	VAX/VMS directory
v1_fixrec	VAX/VMS fixed length record file
v1_gate	Domain/VMS gateway object
v1_undef	VAX/VMS undefined file
v1_varrec	VAX/VMS variable length record file

  The manager automatically converts between Domain and VMS types. For example, if you specify *uasc_$uid* in an **ios_$create** call and specify a VAX/VMS file pathname, a VMS variable-length record file is created. Similarly, if you specify *dir_$uid* and specify a VAX/VMS directory pathname, a VMS directory is created. (See Table 4-3 for details.)

  To create these object types yourself, you need the UIDs for each object type, as follows:

v1_dir	**28FEFAE1.40004234**
v1_fixrec	**298ADA26.B0004234**
v1_gate	**299B51D2.20004234**
v1_undef	**298ADA48.C0004234**
v1_varrec	**298ADA0A.A0004234**

Therefore, to create a VAX/VMS directory object type, you use the following command (in either Aegis or UNIX shells):

```
crty v1_dir -u 28fefae1.40004234
```

- Record seeks are faster than Byte seeks.

- The **ios_$inq_byte_pos** stream call requires significant internal processing to determine the *end of file* location.

- An **ios_$locate** executed on a VMS gateway (*v1_gate*) or VAX/VMS directory (*v1_dir*) is converted into an **ios_$get**.

  An **ios_$locate** with the *ios_$no_rec_bndry_opt* get-put option reads only a single record on *v1_varrec* files, but reads up to 4096 bytes on *v1_fixrec* or *v1_undef* files.

- Domain ASCII file records are terminated by the newline character, while VAX/VMS variable-length records are terminated with a carriage return (CR) character. (The VAX stores records differently than an Apollo workstation. A record length variable at the beginning of the record identifies the length of the record.) The Domain/Access application automatically does the conversion between these two methods of storing records. This conversion is done only for variable records of the type *v1_varrec*.

  Therefore, if you include a newline (NL) character at the end of a variable-length, ASCII record, the VMS system will interpret it as the end of the record. Then, the VMS system will automatically strip off the NL character from the end of the record when you write uasc files to the VAX. When you write the uasc file back to the Apollo workstation, the VAX appends a CR character to the end of the record, which the Domain/Access software strips off and replaces with the NL character again.

- The Domain/Access application does not support many of the **name_$** calls (see Table 4-2). Use the corresponding **ios_$** calls in their place, as indicated in Table 4-5.

Table 4-5. ios_$ Equivalents for name_$ Calls

name_$ Call	ios_$ Call
name_$create file	ios_$create
name _$create_directory	ios_$create
name_$delete_file	ios_$delete
name_$delete_directory	ios_$delete
name_$read_dir	ios_$get

- When you use **ios_$get** to read a VAX/VMS directory, the call returns a 44-character Domain directory record.

- You cannot write to a directory (v1_dir) or VMS gateway (*v1_gate*). However, you can create new files in the directory by using the **ios_$create** call.

Error Status Returns

Table 4-6 explains the meaning of the error status values that can be returned in the **status_$t** parameter of a system call by the Domain/Access software. This table does not include standard TCP/IP, Stream, or other error calls. (See the TCP/IP documentation for information on TCP/IP error messages. See the *Domain System Call Reference* for information on the error status values returned by specific calls.) To receive the error values listed in Table 4-6, you should include one of the following insert files in your program: access.ins.c, access.ins.ftn, access.ins.pas.

NOTE: To contact Customer Services within the United States, call 1-800-227-6556. Outside the U.S., contact your local Apollo office.

Table 4-6. Error Status Returns

Status	Meaning
Errors detected at Apollo node	
access_$chg_attr_close_err	Could not close file to change attributes.
access_$chg_attr_failed	Could not change attributes after the file was closed.
access_$chg_attr_open_err	Could not reopen the file after changing the attributes. Another user may have opened the file for a write operation in the interval.
access_$data_corrupt	The data cache is corrupted. Contact Customer Services.
access_$fatal_error	A fatal error occurred that is not listed here. Contact Customer Services.
access_$gateway_bad	The VMS gateway is not in the expected format. Delete and recreate the gateway; you have an outdated version of the software.
access_$invalid_type	You specified an invalid object type for a set type operation. You must specify a valid v1_xxx type or the equivalent Domain type (such as *uasc*).
access_$key_cache_corrupt	The key cache is corrupted. Contact Customer Services.
access_$pending_failure	The type manager contains an error. Contact Customer Services.
access_$fpi_bad_state	Contact Customer Services.
access_$loopback_fail	The loopback test failed.
access_$protocol_error	A protocol error occurred in the application layer. Contact Customer Services.

Table 4-6. Error Status Returns (Continued)

Status	Meaning
Errors detected at Apollo node (continued)	
access_$bad_connack	A bad message type was received at the time of connection acknowledgement. Contact Customer Services.
access_$conn_ack_to	A timeout occurred at the time of connection acknowledgement.
access_$conn_closed	The connection closed unexpectedly.
access_$connect_to	The connection timed out.
access_$dest_refused	TCP/IP indicated that the destination refused the connection attempt.
access_$extra_data	Unexpected data was received at the end of a TCP/IP message. Contact Customer Services.
access_$host_not_resp	The remote host is not responding.
access_$msg_recv_to	A timeout occurred when data was being received.
access_$msg_send_to	A timeout occurred when data was being sent.
access_$msg_too_long	A message was too long. Contact Customer Services.
access_$string_too_long	The username, filename, password, or host name is too long or has zero length. Filenames must not exceed 252 characters, and all other names must not exceed 64 characters.
access_$unknown_msg_type	A message was received of an unknown type. Contact Customer Services.

Table 4-6. Error Status Returns (Concluded)

Status	Meaning
Errors detected at the VAX	
access_$insufficient_memory	The server on the VAX ran out of dynamic memory. The VAX administrator should check the VAX user quotas.
access_$invalid_arg	An invalid argument was received. Contact Customer Services.
access_$no_such_version	The version of the type manager on the Apollo node does not match the version of the server on the VAX. Make sure that you have installed the most recent Domain/Access software on both systems.
access_$no_such_program	The message received was incomprehensible. Contact Customer Services.
access_$process_died	The VAX server process(es) terminated unexpectedly.
access_$vms_bad_state	Contact Customer Services.
access_$vms_protocol_error	Contact Customer Services.
access_$vs_bad_proc_value	Contact Customer Services.
access_$vs_fatal_error	A fatal error occurred in the VAX server. See "Managing the Domain/Access application" for more information.
access_$login_failed	The login on the VAX system failed. Check the log-in information in your gateway with the **svg** command.

Appendix A

VAX/VMS File Specifications

Standard VAX/VMS file specifications have the following format:

```
node::device:[directory.subdirectory...]file_name.type;version
```

A full VMS file specification with no name components defaulted would therefore look like:

```
taft::dba2:[teletest.test2]com2.ftn;12
```

VAX/VMS file specifications are up to 255 characters long, including all punctuation marks. The punctuation marks separate the file specification's fields. The VMS file system is *not* case sensitive, so you can use uppercase and lowercase letters. (However, you should use lowercase letters when you specify a VAX/VMS file or directory in a Domain pathname.) Table A–1 lists the VAX/VMS file specification components, their meanings, and their default values when using the Domain/Access application.

Table A-1. VAX/VMS File Specification Components

Field	Description	Default
Node	VAX network node name of system where the file is located; up to 6 alphanumeric characters	Determined by the VAX Gateway
Device	Identifies storage device where the file is located; up to 15 characters	Determined by user ID
Directory	A file that identifies a set of files on a disk volume set. The directory name can have one of three formats: 1) A string of up to 39 alphanumeric characters 2) A two-part octal user identification code, 3) A sequence of up to 8 names of type 1, separated by periods (.)	Determined by user ID
File name	A name for this file that is meaningful to you; 1 to 39 alphanumeric characters	None
Type	Identifies the contents of the file, for example as Pascal source code; 1 to 39 characters	None
Version	A decimal number that indicates changes to the file contents; the number increases if you change a file and save it. Range: from 1 to 32767	The highest numbered version

NOTE: VMS file specifications can also include logical names; see the "Logical Names" section later in this appendix.

Note that unlike the Domain file system, VMS files are location sensitive; that is, the file name includes specifications for both the node and the physical device that is being accessed.

The following sections briefly discuss each of the VMS file specification components. They are not all-inclusive; we do not list all standard VAX/VMS device types or file types. For detailed information about VMS files and file specifications, see the VAX/VMS documentation.

Node Names

If the VAX/VMS system that you are connected to is part of a cluster or DECnet network, the different nodes on the network are identified by node names. A **node name** is a one-to-six alphanumeric character string that identifies the processor. The node name must have at least one alphabetic character.

The node name must be followed by double colons (::).

Examples of valid node names include:

```
london::
paris::
```

If a VAX/VMS system does not have a node name, you can indicate the null name by preceding the device name by the double colon, ::.

When you access files through the Domain/Access application, the node is determined by the name in the gateway object that you include in the pathname.

Note that while the VAX/VMS node name and the TCP/IP host name represent a specific VAX machine, they do not have to be the same. For example, it is valid for a VAX/VMS system to have a TCP host name of *london* and a node name of *paris*. The two names serve different purposes. The host name is used by the TCP/IP communications software to establish a connection to the VAX. The node name is used by the VMS record-management system (RMS) software to locate the file.

Device Names

Each physical device that is known to a VAX/VMS system is uniquely identified by its device name. Therefore, the device name in a file definition specifies the physical device that provides or receives the information that you wish to transfer. Disk and tape drives are typical devices. Other valid devices include card readers, communication lines, and consoles.

Device names have the format:

ddcu:

where:

dd	is the Digital-specified code for a particular model or type of physical device
c	is the controller designation, a letter from A through Z. This character identifies the particular controller to which the device is attached.
u	is the unit number, a decimal number in the range 0 through 65535. This value identifies the individual device that is attached to the controller.

You must always follow a device name with a single colon (:).

Typical device names include:

`dma1:`	for an RK06 cartridge disk drive
`msb5:`	for a TS11 magnetic tape drive

When you access files through the Domain/Access application, the initial default device is determined by the user ID specified in your log-in file or in the file pathname. You can set a default device and directory by using the Aegis **wd** or UNIX **cd** command.

When you use telnet, the default device is set when you log in and depends upon your user ID. During the telnet session, you can change your default device with the VMS **set default** command.

Directory Names

A directory is a file that identifies (by names and locations) a set of files on a disk device. Directory names apply only to files on disk devices. A directory name can have any of the following formats:

- A 1-to-39 alphanumeric character string.
- A two-part octal value in user identification code format. This format consists of two octal numbers in the range 0 – 377. You can represent the value either as two 1-to-3 digit values separated by commas, (for example 122,1) or by a 6-digit string (for example 122001). You cannot use the comma format in a sequence of names.
- A sequence of directory names separated by periods, in the format name1.name2.name3. Each name represents a directory level. You can have a maximum of eight directory levels.

You must enclose all directory names in brackets [].

Some valid VMS directory names include:

```
[alf]

[alf.access.test2]

[177,6]

[177006.testfiles.comm.aegis2]
```

When you access files through the Domain/Access application, the initial default directory is determined by the user ID specified in your log-in file or in the file pathname. You can set a default directory by using the Aegis **wd** or UNIX **cd** command.

When you use telnet, the default directory is set when you log in and depends upon your user ID. During the telnet session you can change your default directory with the VMS **set default** command.

File Names

The file name is a 1-to-39 character string that represents the information that is contained in the file. You can use the characters A–Z, a–z, 0–9, _ (underscore), and $ (dollar sign) in file names; however, the dollar sign is reserved for special use. All lowercase characters are interpreted as uppercase. The combination of file name, file type, and version number uniquely identifies any file within a directory.

Some valid file names include:

```
testfile
access
```

File Types

A file type is an optional 1–to–39 character string that identifies the type of information contained in the file. File types are not required; a file can have a null file type. Valid characters in file types are A–Z, a–z, 0–9, _ (underscore), and $ (dollar sign); however, the dollar sign is reserved for special use. All lowercase characters are interpreted as uppercase.

By convention, VAX/VMS uses a set of standard file types to identify specific classifications of files and to provide default file types in many commands. Some typical file types include the following:

c	Input source for the C compiler
com	Indirect command file
dat	Input or output data file
dir	Directory file
lis	Listing file created by a compiler or assembler
obj	Object file created by a compiler or assembler
pas	Pascal source code file
txt	Input file for text libraries or MAIL command output

When you access files through the Domain/Access application, there is no default file type. When you use telnet, the default file type depends upon the VMS command that you enter. A complete list of VMS default file types is located in the VAX/VMS documentation.

File types must be preceded by a period (.).

File Version Numbers

Version numbers are decimal numbers from 1 to 32767 that differentiate versions of a file. When you update or modify a file and do not specify a version number for the output file, VMS:

1. Saves the original version for backup
2. Increments the version number by one
3. Saves the changed file using this version number

The Domain/Access software supports VMS version–number incrementing. Therefore, if you edit or otherwise modify a VMS file using the Domain/Access application, the resulting file will have an incremented version number.

Whenever you access a file, for example to print it, the default version number is the highest number present (in other words, the most recent version).

Logical Names

A logical name is a user– or system–defined name, or alias, for:

- All or part of a file specification
- A physical device
- A user name

You can use logical names as a shorthand way of specifying files or directories that you refer to frequently. For example, you might assign the logical name *home* to your VMS login default disk and directory. You can also use logical names in file specifications to keep your programs and command procedures independent of physical file specifications.

The VAX/VMS system includes predefined logical names for commonly used system elements. These include the *sys$system* directory, which contains operating system programs and procedures, the *sys$manager* directory which contains the system manager files, and *sys$sysdevice*, the system disk that contains the system directories. VAX/VMS system–defined logical names have the format xxx$name, where xxx defines the system component that uses the logical name.

You create logical names by using the VMS DCL commands **define** and **assign**. The VAX/VMS DCL documentation describes logical names in detail and tells you how to manage them. You must be logged in to the VAX/VMS system to use DCL commands. Therefore, you must use telnet to manage logical names from an Apollo workstation. You *can* use logical names in Domain/Access pathnames.

When you use a logical name as part of a file specification, the logical name must:

- Be the leftmost component of the VMS part of the file specification. When using the Domain/Access application, it must follow the name of the gateway object.

- End in a colon.

For example, if you assign the logical name *taxes* to *use1:[income.taxes]*, you can refer to the file *use1:[income.taxes]july.dat* as *taxes:july.dat*. To list the *taxes* directory, using the gateway object *v_paris*, you enter:

```
in a UNIX shell
ls //node_a/v_paris/taxes:

in an Aegis shell
ld  //node_a/v_paris/taxes:
```


Glossary

This glossary contains definitions from *The Digital Dictionary, A Guide to Digital Equipment Corporation's Technical Terminology*, Second Edition, Copyright 1986. These definitions are indicated by an asterisk (*).

absolute pathname — A pathname that begins at the Domain root directory, the // directory.

account* — A key to the system and a unit of system accounting. Each system user, including parts of the system itself, has an account. The system manager creates these accounts in the master file directory and assigns an account number to each. Individual users are identified by an account name as well. When you log in, you log in under a particular account name or number. This number informs the system where your files are and what kind of access to other files and system facilities you should be given.

account name* — A string that identifies a particular account used to accumulate data on a job's resource use. This name is the user's accounting charge number, not the user's user identification code (UIC).

address — An identifier applied to a host or gateway that is unique over the network or internet. Each network has its own address format.

connection — A logical communication path established between two endpoints.

DECnet*

1. The DIGITAL software facility that enables a user to access information on a remote computer via telecommunications lines.

2. DIGITAL networking software that runs on nodes in both local and wide-area networks.

device name*

A field in a **file specification** that identifies the device unit on which a file is stored. Device names also include the mnemonics that identify an I/O peripheral device in a data transfer request. A device name consists of a mnemonic followed by a controller identification letter (if applicable), followed by a unit number (if applicable), and ends with a colon (:).

directory

1. On VAX/VMS systems*: A file that briefly catalogs a set of files stored on disk or tape. The directory includes the name, type, and version number of each file in the set, as well as a unique number that identifies the file's actual location and points to a list of its attributes.

2. On Domain systems: An object that contains information about objects beneath it in the naming tree.

directory name*

The field in a **file specification** that identifies the directory in which the file is listed. It begins with a left bracket or left angle bracket ([or <) and ends with a right bracket or right angle bracket (] or >). The brackets enclose either a group number and a user number separated by a comma, or an alphanumeric directory list.

Domain

In this manual, we use the term Domain to include both Aegis and Domain/IX operating systems.

Domain/Access

A product of Apollo Computer Inc. that allows Domain nodes to access files on Digital Equipment Corporation VAX computers that run the VMS or MicroVMS operating systems.

ETHERNET network

A 10 mbs local area network developed by Digital Equipment Corporation, Intel, and Xerox Corporation. The network transmits 10 megabits per second on coaxial cable with CSMA/CD baseband signaling.

filename

A name that represents a file or directory. We use this term to represent a file system identifier on *either* a Domain or a VAX/VMS system. *See also* File Name, File Specification, and Pathname.

file name*

The field preceding a file type in a **file specification** that contains a 1- through 39-character name for a file.

file specification*

A unique name for a file on a mass storage medium. It identifies the node, the **device**, the **directory name**, the **file name**, the **file type**, and the **version number** under which a file is stored.

File Transfer Protocol

(FTP) A protocol for transferring files between host computers defined by the Defense Advanced Research Projects Agency. FTP uses TCP and IP as underlying protocols.

file type*

The field in a **file specification** that consists of a period (.) followed by a 0- to 39-character type identification, by convention, this field identifies the generic class of files that have the same use or characteristics, such as compiler and assembler listing files, binary object files, and so on.

FTP

See File Transfer Protocol

gateway object — A Domain object (**v1_gate**) containing connection information that the Domain/Access type manager needs to establish a TCP/IP connection with the VAX.

gateway node — A node containing a gateway server, which is a process that receives a packet from one network, translates it to the protocols of another network, and transmits it on that network.

host — An entity such as a computer or workstation that communicates over a network. A host can both initiate communications and respond to communications that are addressed to it.

indexed file* — A file that contains records and a primary key index (and optionally one or more alternate key indices) used to process the records sequentially by index or randomly by index.

Internet — 1. Two or more possibly dissimilar networks that are connected together. (The initial i in internet is lowercase in this context.)

2. An internet that conforms to a set of protocols defined by the Defense Advanced Research Projects Agency that includes the Internet Protocol and Transmission Control Protocol. Also called a DARPA Internet. (The initial I in Internet is always uppercase in this context.)

Internet address — 1. An address that conforms to the DARPA–defined **Internet** protocol. A unique, 4–byte number that identifies a host or gateway on the Internet, consisting of a **network number** followed by a **local address**. Internet addresses are usually expressed as four decimal numbers in the range 0–255, separated by periods.

2. An address that uniquely identifies a destination on an **internet**.

Internet gateway — A device or set of devices that connects between two unlike networks, such as Domain Ring and ETHERNET. A gateway can route packets and perform protocol translation services.

Internet name — A mnemonic identifier applied to an Internet host, gateway, or network. Internet names are mapped into Internet addresses.

LAN — *See* Local Area Network

link — A Domain object that points from one place in the naming tree to another. A link contains the pathname of another object.

local address — An address that uniquely identifies a destination within a single network.

Local Area Network — (LAN) A communications network linking a number of devices that are located within a relatively short distance, typically less than a mile.

logical name* — A user–specified name for any portion of all of a **file specification**. For example, the logical name INPUT can be assigned to a terminal device from which a program reads data entered by user. Logical name assignments are maintained in logical name tables for each process, each group, and the system.

network — A set of entities, such as computers, terminals, and workstations, that communicate with each other over some physical medium. Also used to refer to the medium and the protocols that specify how data is communicated between the entities.

node — 1. A VAX/VMS system.
2. A Domain workstation or server processor.

node ID — The Domain node identification number.

node name — 1. The VAX/VMS node identifier; a 1–6 character string.
2. The name that identifies a Domain node on its network.

node specification — A Domain node identifier, the node's node ID, address on the Domain network, or node name.

object — A storage container that is defined by the set of operations that can be performed on it. Domain objects include files, directories, processes, and VAX/VMS gateway objects.

pathname — On Domain systems, a series of names separated by slashes that describe the path the operating system must take to get from some starting point in the network to a destination object. For the purposes of Domain/Access, a Domain pathname is the equivalent of a VMS file specification.

port — A number that represents a destination, within a TCP/IP host, for a message or connection.

protocol — A set of rules that governs the procedures used in exchanging information between two entities such as computers, workstations, and terminals.

relative pathname — A pathname that does not start at the Domain root directory (//). For example, a pathname beginning at the working directory.

TCP/IP — Transmission Control Protocol / Internet Protocol. An acronym used to refer to the TCP and IP protocols and related Internet protocols, such as FTP and Telnet, defined by the Defense Advanced Projects Agency.

telnet — A remote terminal emulation protocol defined by the Defense Advanced Research Projects Agency for internetwork communications. Telnet uses TCP and IP as underlying protocols.

type manager — A Domain object that translates Domain object types and record formats into those of a foreign file system, and vice versa.

user ID or user name — The name by which a system identifies a particular user. You must specify a user ID and password to gain access to a Domain or VMS system.

VAX* — Virtual address extension.

VAX/VMS* — Virtual address extension/virtual memory system.

virtual address extension (VAX)* — A computer made by the Digital Equipment Corporation. VAX is a high-performance, multiprogramming computer system based on a 32-bit architecture.

virtual address extension/virtual memory system (VAX/VMS)* — VAX/VMS is the virtual memory operating system that runs on the entire range of VAX (32-bit) processors.

v1_dir — The Domain identifier for a VAX/VMS directory.

v1_fixrec The Domain identifier for a VAX/VMS fixed length record file.

v1_gate The Domain object that represents a gateway object.

v1_undef The Domain identifier for a VAX/VMS undefined file.

v1_varrec The Domain identifier for a VAX/VMS variable length record file.

version number* The field following the file type in a file specification. It begins with a semicolon (;) or period (.) and is followed by a number which generally identifies it as the latest file created of all files having the identical file specification but for version number.

Index

Primary page references are listed first. The letter *f* means "and the following page"; the letters *ff* mean "and the following pages". Symbols are listed at the beginning of the index. Task-oriented entries appear in color.

Symbols

A

B

C

D

E

F